Lucrarea 2: Constructori şi destructori în limbajul C++

Ce ne propunem astăzi

Azi vom învăţa cum să creăm şi să distrugem obiecte folosind caracteristicile limbajului C++. În general obiectele necesită iniţializare înainte de a fi folosite, de exemplu, avem nevoie să iniţializăm elementele unei matrici de obiecte cu valoarea zero. Până acum acest lucru era posibil prin utilizarea unei funcţii de iniţializare. Deoarece această necesitate de iniţializare a unor elemente este atât de des întâlnită, limbajul C++ permite iniţializarea obiectelor atunci când acestea sunt create. Această iniţializare este făcută automat prin intermediul constructorilor. Pentru a distruge un obiect creat vom folosi opusul constructorului, şi anume destructorii.

Constructorii

O funcţie **constructor** este o funcţie specială, care este membru al unei clase şi are *acelaşi nume* cu clasa din care face parte. Funcţia constructor este apelată automat când este creată o nouă instanţă (adică un nou obiect) al clasei respective - şi numai atunci. Funcţia constructor nu returnează nici un rezultat şi poate avea oricâţi parametri. Mai multe detalii se pot găsi în referinţa [1] la pagina 36.

Constructorii sunt metode speciale care folosesc la crearea si iniţializarea instanţelor unei clase şi au următoarele proprietăţi:

- pot fi creaţi de către programator;
- în absenţa constructorilor definiţi, clasei se ataşează în mod implicit un constructor. Un astfel de constructor se numeşte **constructor implicit** (constructori fără parametri generaţi automat de limbaj dacă programatorul nu a definit unul)**;**
- constructorii impliciţi nu se generează în cazul în care clasa are creat un alt constructor;
- constructorii au acelaşi nume ca şi clasa căreia îi aparţin;
- constructorii sunt apelaţi de fiecare dată când se creează noi instanţe ale clasei;
- o clasă poate avea mai mulţi constructori creaţi de programator cu număr şi tip diferit de parametrii.

Destructorii

Destructorul are un comportament opus constructorului. Acesta este apelat automat în momentul în care un obiect este eliberat din memorie, fie pentru că existenţa obiectului s-a încheiat, fie pentru că obiectul a fost alocat dinamic (cu *new*) şi este eliberat utilizând operatorul *delete*.

Destructorul are întotdeauna acelaşi nume ca şi clasa şi este precedat de tilda (~) ca un prefix. La fel ca şi în cazul constructorului, destructorul nu returnează nici o valoare. Destructorul este folosit pentru a elibera memoria alocată unui obiect în momentul în care acesta a fost iniţializat.

Mai multe detalii despre destructori se pot găsi în referinţa [1] la pagina 36.

Supraîncărcarea constructorilor

Constructorul poate fi supraîncărcat cu oricâte funcţii care au acelaşi nume, dar cu număr şi tip diferit de parametrii. Compilatorul va apela acea funcţie care se potriveşte atât la numărul de parametrii, cât şi la tipul acestora.

În cazul în care o clasă nu are nici un constructor, compilatorul va crea următorii constructori ca şi funcţii membru:

- *constructorul vid* (***empty constructor***) - un constructor care nu are parametri şi are un bloc vid de instrucţiuni şi în consecinţă, nu face nimic;
- *constructorul de copiere* (***copy constructor***) - un constructor care preia un singur parametru, o referinţă către un obiect de aceeaşi clasă, pentru ca astfel fiecare membru al obiectului transmis să fie copiat în noul obiect.

Mai multe detalii despre supraîncărcarea constructorilor se pot găsi în referința [1] la pagina 114.

Pointeri către obiecte

În limbajul C++ se pot crea pointeri care să indice către obiecte. Pentru a accesa un membru al obiectului se va folosi operatorul ->. Mai multe detalii se pot găsi în referința [1] la pagina 84.

Partea practică. Mod de lucru

Iată paşii care trebuie urmaţi pentru dezvoltarea cu succes a programelor:

1. În Anexa 2 exemplul 1 este reluată problema din Anexa 1.1 exemplu 5. În acest exemplu funcţia de *initializar*e() a fost înlocuită de constructorul clasei. Parametrii sunt transmişi constructorului în momentul în care clasa este creată.
2. În Anexa 2 exemplu 2 este prezentat un program care conţine clasa *Persoana* cu următoarele variabile: nume, prenume, vârsta. Datele sunt iniţializate prin intermediul constructorului, după care sunt afişate pe ecran. În plus faţă de exemplul anterior aveţi iniţializarea în constructor a unor variabile de tip şir de caractere.
3. ***După exemplul prezentat anterior să se realizeze un program care conţine o clasă denumită Carte cu următoarele variabile: titlu, nume autori, editură, an publicare. Programul va trebui să permită adăugarea unor înregistrări şi afişarea acestora. Iniţializarea variabilelor se va face prin intermediul constructorului.***
4. Vom studia exemplele următoare şi vom rezolva cerinţele propuse:
    - În Anexa 2 exemplu 3 este prezentată folosirea destructorului *~Dreptunghi()* pentru a elibera memoria alocată pentru cele două lungimi de laturi ale dreptunghiului, având în vedere că acestea au fost alocate dinamic cu operatorul *new*. În exemplul acesta a fost folosită apelarea explicită a destructorului.
    - În Anexa 2 exemplu 4 este folosit destructorul *~Persoana()* pentru a elibera memoria alocată pentru numele şi prenumele persoanei, având în vedere că acestea au fost alocate dinamic cu operatorul *new*. În acest caz apelarea destructorului este implicită.
    - În Anexa 2 exemplu 5 este prezentat un program care utilizează constructorii şi destructorii pentru afişarea informaţiilor despre un student. Paşii efectuaţi de către compilator sunt prezentaţi în următoarele rânduri:
        - pornirea procesului
        - se creează un obiect *s* de tip *Student* moment în care se apelează constructorul (*Student()*) clasei Student, iar în consolă se afişează mesajul scris în constructor: ” Constructor: Informatiile despre student:”

- introducerea datelor studentului prin apelarea funcţiei citire_date().
- afişarea datelor prin apelarea funcţiei afişare_date()
- înainte de a se termina execuţia programului se apelează destructorul(~*Student()*) , acest lucru va duce la afişarea mesajului: ”Destructor: Eliberare memorie! ”

- În Anexa 2 exemplu 6 este prezentat un program care foloseşte pointeri la membri.

5. ***După exemplul 5 prezentat în Anexa 2 să se modifice programul astfel încât iniţierea atributelor unui obiect de tip Student (nume, rol, adresa, cod postal) să se facă în constructor. Să se creeze şi un destructor care se va apela explicit. Constructorul va primi ca parametri cele patru valori.***
6. În Anexa 2 exemplu 7 este prezentat un program care foloseşte supraîncărcarea constructorilor. Constructorii se apelează în momentul în care se creează un nou obiect, iar lipsa sau numărul şi tipul parametrilor decid care din constructor se va apela. În exemplu obiectul *dreptb* a fost declarat fără parametri, ceea ce a condus la iniţializarea cu ajutorul constructorului fără parametrii, care atribuie ambelor variabile membru valoarea 5.
7. ***După exemplul 7 prezentat în Anexa 2 să se creeze o clasa Profesor care are ca şi variabile: nume, departament, grad didactic, vechime. Programul va permite adăugarea de noi profesori si afişarea informaţiilor despre aceştia.***

Cu ce ne-am ales?

Prin programele prezentate ca exemple şi problemele propuse de la partea practică a laboratorului am reuşit să ne familiarizăm cu folosirea constructorilor şi destructorilor, descoperind rolul acestora în programarea orientată pe obiecte. Constructorii şi destructorii se utilizează în toate programele orientate pe obiecte.